SON-1661 PA NT APPLICATION #4

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the Patent Application of)
)
YOSHIHIKO IMAMURA)
) ATTN:
Serial No. (Not yet assigned)) APPLICATION BRANCH
)
Filed: October 19, 1999)
)
For: PARALLEL PROCESSOR, PARALLEL)
PROCESSING METHOD AND STORING MEDIUM)

CLAIM TO PRIORITY UNDER 35 U.S.C. 119

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

The benefit of the filing date of the following prior applications filed in the following foreign country is hereby requested and the right of priority provided under 35 U.S.C. 119 is hereby claimed:

Japanese Patent Appl. No. P10-302679, filed October 23, 1998

In support of this claim, filed herewith is a certified copy of said original foreign applications.

Respectfully submitted,

Dated: October 19, 1999

Ronald P. Kananen
Reg. No. 24,104

RADER, FISHMAN & GRAUER P.L.L.C.
1233 20TH Street, NW
Suite 501
Washington, DC 20036
202-955-3750-Phone
202-955-3751 - Fax

S99P1175 U500

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: １９９８年１０月２３日

出願番号
Application Number: 平成１０年特許願第３０２６７９号

出願人
Applicant (s): ソニー株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

１９９９年　８月１８日

特許庁長官
Commissioner,
Patent Office 伊佐山建

出証番号　出証特平１１－３０５７９１９

【書類名】　特許願

【整理番号】　9800521607

【提出日】　平成10年10月23日

【あて先】　特許庁長官殿

【国際特許分類】　G06F　5/00

【発明の名称】　並列処理装置、並列処理方法および記録媒体

【請求項の数】　22

【発明者】

【住所又は居所】　東京都品川区北品川６丁目７番３５号　ソニー株式会社内

【氏名】　今村　義彦

【特許出願人】

【識別番号】　000002185

【氏名又は名称】　ソニー株式会社

【代表者】　出井　伸之

【代理人】

【識別番号】　100094053

【弁理士】

【氏名又は名称】　佐藤　隆久

【手数料の表示】

【予納台帳番号】　014890

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9707389

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　並列処理装置、並列処理方法および記録媒体

【特許請求の範囲】

【請求項１】

プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置において、

一の前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記他の処理手段は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する

並列処理装置。

【請求項２】

前記他の処理手段は、同期待ち命令を実行して同期待ち状態になり、前記一の処理手段による前記同期待ち命令に対応する前記待機命令の実行に基づいて前記同期待ち状態を解除する

請求項１に記載の並列処理装置。

【請求項３】

前記プログラムを記憶する第１の記憶手段と、

前記複数の処理手段にそれぞれ対応して設けられ、対応する処理手段で実行されるプログラムを前記第１の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第１の記憶手段に比べてアクセス速度が速い第２の記憶手段と

をさらに有し、

前記第２の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムの記憶を継続する

請求項１に記載の並列処理装置。

【請求項４】

前記第２の記憶手段は、対応する前記処理手段がプログラムの終了を示すプログラム終了命令を実行するまで、前記プログラムの記憶を継続する

請求項３に記載の並列処理装置。

【請求項５】

前記処理手段は、前記第１の記憶手段に記憶されているプログラムを他の前記処理手段で実行させることを指示するプログラム実行指示命令を実行し、

当該他の処理手段に対応する前記第２の記憶手段は、前記プログラム実行指示命令によって実行が指示されたプログラムを前記第１の記憶手段から読み込んで記憶する

請求項１に記載の並列処理装置。

【請求項６】

前記処理手段による前記プログラム実行指示命令に基づいて、当該プログラム実行指示命令によって実行が指示されたプログラムを実行させる前記処理手段を決定し、当該プログラム実行指示命令によって実行が指示されたプログラムを前記第１の記憶手段から前記決定した処理手段に対応する前記第２の記憶手段に読み込むプログラム実行割り付け手段

をさらに有する請求項５に記載の並列処理装置。

【請求項７】

前記プログラム実行指示命令によって実行が指示されたプログラムに記述された前記待機命令に基づいて前記他の処理手段が待機状態になったときに、前記プログラム実行指示命令を実行した前記処理手段が前記同期解除命令を実行する

請求項５に記載の並列処理装置。

【請求項８】

前記プログラム実行指示命令によって実行が指示されたプログラムに記述された前記待機命令に基づいて前記他の処理手段が待機状態になったときに、前記プログラム実行指示命令を実行した前記処理手段以外の前記処理手段が前記同期解除命令を実行する

請求項５に記載の並列処理装置。

【請求項９】

前記複数の処理手段と、

当該複数の処理手段を接続する共有バスと

が単体の半導体チップ内に組み込まれている

請求項１に記載の並列処理装置。

【請求項１０】

プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置において、

一の前記処理手段は、待機命令を実行すると前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記他の処理手段は、前記待機解除命令を実行したときに、前記一の処理手段が前記待機状態になっていない場合に、前記一の処理手段が前記待機状態になるまで同期待ち状態になる

並列処理装置。

【請求項１１】

前記他の処理手段は、前記待機解除命令を実行したときに、前記一の処理手段が前記待機状態になっている場合に、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する

請求項１０に記載の並列処理装置。

【請求項１２】

前記プログラムを記憶する第１の記憶手段と、

前記複数の処理手段にそれぞれ対応して設けられ、対応する処理手段で実行されるプログラムを前記第１の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第１の記憶手段に比べてアクセス速度が速い第２の記憶手段と

をさらに有し、

前記第２の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムの記憶を継続

する

請求項１０に記載の並列処理装置。

【請求項１３】

前記第２の記憶手段は、対応する前記処理手段がプログラムの終了を示すプログラム終了命令を実行するまで、前記プログラムの記憶を継続する

請求項１２に記載の並列処理装置。

【請求項１４】

プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段と、

前記プログラムを記憶する第１の記憶手段と、

前記複数の処理手段にそれぞれ対応して設けられ、対応する処理手段で実行されるプログラムを前記第１の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第１の記憶手段に比べてアクセス速度が速い第２の記憶手段と

を有する並列処理装置において、

前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記第２の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムの記憶を継続する

並列処理装置。

【請求項１５】

前記第２の記憶手段は、対応する前記処理手段がプログラムの終了を示すプログラム終了命令を実行するまで、前記プログラムの記憶を継続する

請求項１４に記載の並列処理装置。

【請求項１６】

前記処理手段は、前記第１の記憶手段に記憶されているプログラムを他の前記処理手段で実行させることを指示するプログラム実行指示命令を実行し、

当該他の処理手段に対応する前記第２の記憶手段は、前記プログラム実行指示命令によって実行が指示されたプログラムを前記第１の記憶手段から読み込んで記憶する

請求項１４に記載の並列処理装置。

【請求項１７】

前記処理手段による前記プログラム実行指示命令に基づいて、当該プログラム実行指示命令によって実行が指示されたプログラムを実行させる前記処理手段を決定し、当該プログラム実行指示命令によって実行が指示されたプログラムを前記第１の記憶手段から前記決定した処理手段に対応する前記第２の記憶手段に読み込むプログラム実行割り付け手段

をさらに有する請求項１６に記載の並列処理装置。

【請求項１８】

プログラムに記述された命令に基づいて少なくとも第１の処理および第２の処理を相互に並列に行う並列処理方法において、

前記第１の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記第２の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する

並列処理方法。

【請求項１９】

前記第２の処理は、同期待ち命令の実行によって同期待ち状態になり、前記第１の処理における前記同期待ち命令に対応する前記待機命令の実行に基づいて前記同期待ち状態を解除する

請求項１８に記載の並列処理方法。

【請求項２０】

プログラムに記述された命令に基づいて少なくとも第１の処理および第２の処理を相互に並列に行う並列処理方法において、

前記第１の処理は、待機命令の実行によって前記プログラムに基づいた処理を

中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記第２の処理は、前記待機解除命令の実行時に、前記第１の処理が前記待機状態になっていない場合に、前記第１の処理が前記待機状態になるまで同期待ち状態になる

並列処理方法。

【請求項２１】

プログラムに記述された命令に基づいて相互に並列に行われる第１の処理および第２の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって、

前記第１の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、

前記第２の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する処理である

記録媒体。

【請求項２２】

プログラムに記述された命令に基づいて相互に並列に行われる第１の処理および第２の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって、

前記第１の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、

前記第２の処理は、前記待機解除命令の実行時に、前記第１の処理が前記待機状態になっていない場合に、前記第１の処理が前記待機状態になるまで同期待ち状態になる処理である

記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、並列処理装置、並列処理方法および当該方法の手順をコンピュータで読み取り可能に記録した記録媒体に関する。

【０００２】

【従来の技術】

例えば、Ｕｎｉｘ（商標名）などのＯＳ(Operating System)を搭載したシングルプロセッサでは、マルチタスク環境でのプログラムの実行に際してローカルメモリに同時に存在する複数個のプログラムの進行を管理する機能が必要であり、当該機能において「プログラム」という言葉を明確な対象概念とするために「プロセス」という概念が用いられている。プロセスは、独立したプログラムの実行単位であり、当該プログラムが独自にアクセスできるメモリ空間（ユーザメモリ空間）をローカルメモリに設定する。また、プログラムの実行はプロセスの起動を意味し、プログラムの終了はプロセスの消滅を意味する。また、プロセスは、他のプロセスを起動および消滅させたり、他のプロセスと通信を行うことができる。

【０００３】

ところで、シングルプロセッサでは、ＣＰＵ(Central Processing Unit) が１個であるため、起動できるプロセスは、任意の時刻で最大１個である。従って、シングルプロセッサでは、独立した複数のプログラムにユーザメモリ空間を同時に割り当て、時分割(Time Sharing)方式で当該複数のプログラムを交互に実行して複数のプロセスを交互に起動することで、マルチタスク環境を実現している。

このとき、一のプロセスが起動状態にある場合に、他のプロセスは待機状態になっている。

【０００４】

上述したマルチタスク環境下では、複数のプロセス相互間でのメッセージ通信を以下に示すよう行う。

すなわち、シングルプロセッサでは、前述したように、任意の時刻で起動状態

になるプロセスは最大１個であるため、メッセージを送信する一のプロセスが起動状態にあるときには、当該メッセージを受信する他のプロセスは待機状態になっている。従って、メッセージを送信する起動状態のプロセスは、Ｏ／Ｓのカーネル内のプロセス管理タスクを呼び出し、当該メッセージを受信するプロセスが前回待機状態に遷移する直前の起動状態を記憶しているメモリ上のテーブル（通常、スレッドのコンテキストを保存しているテーブル）にメッセージを送信する旨を書き込む。そして、メッセージ受信するプロセスが、次に起動状態に遷移すると、当該テーブルを参照することで、メッセージを受信したことを知り、それに応じた処理を行う。一方、例えば、メッセージを受信するプロセスが当該メッセージの受信を条件に次の処理に進むものであり、起動状態に遷移してテーブルを参照したときにメッセージの受信の事実が無いと判断した場合には、当該プロセスは待機状態になる。そして、当該プロセスは、メッセージの受信を確認した後に、起動状態に遷移する。

【０００５】

一方、例えば共有バスを介して接続された複数のＣＰＵが相互に独立した複数のプログラムを並列に実行するマルチプロセッサでは、通常、１個のＣＰＵにおいて任意の時刻で最大１個のプロセスが起動状態となり、相互に異なるＣＰＵ上で複数のプロセスが同時に起動状態となり得る。

そして、プロセス相互間での通信は、ユーザプログラム（アプリケーションプログラム）内に明示された命令コードに基づいて、例えば、送信側のプロセスが共有バスにメッセージを流し、共有バスを監視しているアービタが受信側のプロセスに当該メッセージを通知して行われる。従って、プロセス相互間でメッセージ通信を実現するには、メッセージの送信側のプロセスと受信側のプロセスとの双方が起動状態である必要がある。

このように、通常、マルチプロセッサでは、前述したシングルプロセッサのようなプロセス管理タスクを用いたメッセージ通信は行われていない。すなわち、マルチプロセッサでは、プロセス管理タスクというものが存在しない。

【０００６】

ところで、マルチプロセッサでは、並列に動作する複数のプロセス相互間で同

期をとる必要がある場合に、上述したメッセージ通信を用いて同期を実現している。

以下、マルチプロセッサにおける従来のプロセス相互間の同期方法について説明する。

先ず、一般的なマルチプロセッサの構成について説明する。

図５は、一般的なマルチプロセッサ１の構成図である。

図５に示すように、マルチプロセッサ１は、共有バス１７を介して例えば４個のプロセッサエレメント11_1～11_4を接続した構成をしている。また、共有バス１７には、共有メモリ１５およびアービタ１６が接続されている。

【０００７】

ここで、プロセッサエレメント11_1は、例えば、図６に示すように、プロセッサコア３１およびローカルメモリ３２を有し、共有バス１７を介して共有メモリ１５から読み出したユーザプログラムをローカルメモリ３２に記憶し、ローカルメモリ３２に記憶しているユーザプログラムの命令コードをプロセッサコア３１に順次に供給して実行する。プロセッサエレメント11_2～11_4は、例えば、プロセッサエレメント11_1と同じ構成をしている。

【０００８】

また、アービタ１６は、プロセッサエレメント11_1～11_4の実行状態（処理の負荷など）を監視し、当該実行状態に基づいて、共有メモリ１５に記憶されているソフトウェアリソースをハードウェアリソースであるプロセッサエレメント11_1～11_4に割り当てる。具体的には、アービタ１６は、共有メモリ１５に記憶されているユーザプログラムをプロセッサエレメント11_1～11_4の図６に示すローカルメモリ３２に読み込む。

アービタ１６は、例えば、図７に示すように、ユーザプログラムであるメインプログラムＰｒｇ＿Ａおよびサブプログラムプログラム Ｐｒｇ＿Ｂ，Ｐｒｇ＿Ｃ，Ｐｒｇ＿Ｄ，Ｐｒｇ＿Ｅを、同時刻あるいは異なる時刻に、図７中矢印で示すプロセッサエレメント11_1～11_4のローカルメモリ３２に読み込む。

【０００９】

次に、図５に示すマルチプロセッサ１における従来のプログラム（プロセス）

相互間での同期方法について説明する。

先ず、共有メモリ１５に記憶されているメインプログラムＰｒｇ＿Ａが、アービタ１６によって、プロセッサエレメント11_1のローカルメモリ３２に読み込まれ、図８に示すように、メインプログラムＰｒｇ＿Ａに記述された命令コードがプロセッサエレメント11_1で順次に実行される。

そして、プロセッサエレメント11_1において、命令コード「ｇｅｎ（Ｐｒｇ＿Ｂ）」が実行されると、その旨を示すメッセージが共有バス１７を介してアービタ１６に通知される。そして、アービタ１６によって、プロセッサエレメント11_1～11_4の実行状態に基づいて、共有メモリ１５に記憶されているサブプログラムＰｒｇ＿Ｂが、プロセッサエレメント11_2のローカルメモリ３２に読み込まれ、サブプログラムＰｒｇ＿Ｂに記述された命令コードがプロセッサエレメント11_2で順次に実行される。

【００１０】

その後、プロセッサエレメント11_1において、命令コード「ｇｅｎ（Ｐｒｇ＿Ｃ）」が実行されると、その旨を示すメッセージが共有バス１７を介してアービタ１６に通知される。そして、アービタ１６によって、プロセッサエレメント11_1～11_4の実行状態に基づいて、共有メモリ１５に記憶されているサブプログラムＰｒｇ＿Ｃが、プロセッサエレメント11_3のローカルメモリ３２に読み込まれ、サブプログラムＰｒｇ＿Ｃに記述された命令コードがプロセッサエレメント11_3で順次に実行される。

【００１１】

その後、プロセッサエレメント11_1において、命令コード「ｇｅｎ（Ｐｒｇ＿Ｄ）」が実行されると、その旨を示すメッセージが共有バス１７を介してアービタ１６に通知される。そして、アービタ１６によって、プロセッサエレメント11_1～11_4の実行状態に基づいて、共有メモリ１５に記憶されているサブプログラムＰｒｇ＿Ｄが、プロセッサエレメント11_4のローカルメモリ３２に読み込まれ、サブプログラムＰｒｇ＿Ｄに記述された命令コードがプロセッサエレメント11_4で順次に実行される。

【００１２】

その後、プロセッサエレメント11_1において、命令コード「ｗａｉｔ（Ｐｒｇ＿Ｄ」が実行されると、プロセッサエレメント11_1の処理が同期待ち状態になる。

【００１３】

その後、プロセッサエレメント11_4において、サブプログラムＰｒｇ＿Ｄの最後の命令コード「ｅｎｄ」が実行されると、サブプログラムＰｒｇ＿Ｄの終了を示すメッセージが、例えばアービタ１６を介してプロセッサエレメント11_1に通知される。これにより、プロセッサエレメント11_1は、同期待ち状態を解除して、次の命令コードを実行する。

【００１４】

その後、プロセッサエレメント11_1において、命令コード「ｗａｉｔ（Ｐｒｇ＿Ｃ」が実行されると、プロセッサエレメント11_1の処理が同期待ち状態になる。

【００１５】

その後、プロセッサエレメント11_3において、サブプログラムＰｒｇ＿Ｃの最後の命令コード「ｅｎｄ」が実行されると、サブプログラムＰｒｇ＿Ｃの終了を示すメッセージが、例えばアービタ１６を介してプロセッサエレメント11_1に通知される。これにより、プロセッサエレメント11_1は、同期待ち状態を解除して、次の命令コードを実行する。

【００１６】

その後、プロセッサエレメント11_1において、命令コード「ｇｅｎ（Ｐｒｇ＿Ｅ）」が実行されると、その旨を示すメッセージが共有バス１７を介してアービタ１６に通知される。そして、アービタ１６によって、プロセッサエレメント11_1～11_4の実行状態に基づいて、共有メモリ１５に記憶されているサブプログラムＰｒｇ＿Ｅが、例えばプロセッサエレメント11_4のローカルメモリ３２に読み込まれ、サブプログラムＰｒｇ＿Ｃに記述された命令コードがプロセッサエレメント11_4で順次に実行される。

【００１７】

その後、プロセッサエレメント11_1において、再び、命令コード「ｇｅｎ（Ｐｒｇ＿Ｄ）」が実行されると、その旨を示すメッセージが共有バス１７を介してアービタ１６に通知される。そして、アービタ１６によって、プロセッサエレメント11_1～11_4の実行状態に基づいて、共有メモリ１５に記憶されているサブプログラムＰｒｇ＿Ｄが、プロセッサエレメント11_3のローカルメモリ３２に読み込まれ、サブプログラムＰｒｇ＿Ｄに記述された命令コードがプロセッサエレメント11_3で順次に実行される。

【００１８】

【発明が解決しようとする課題】

上述したように、従来のマルチプロセッサ１では、異なるプロセッサエレメント上で実行されるプログラム（プロセス）相互間での同期は、一のプロセッサエレメントによる命令コード「ｗａｉｔ」の実行で発生した同期待ち状態を、他のプロセッサエレメントにおけるプログラムの実行終了を示す命令コード「ｅｎｄ」の実行に基づいて解除するという単純なものである。

すなわち、一のプログラムに基づいたプロセッサエレメントの同期待ち状態は、他のプロセッサエレメントおけるプログラムの実行が終了しなれば解除できない。従って、異なるプロセッサエレメント上で実行する異なるプログラム相互間で、例えば、プログラムの途中に記述された命令コード相互間で同期をとるなどの多様な形態の同期を実現することができないという問題がある。

【００１９】

また、上述した実施形態では、アービタ１６は、例えば、プロセッサエレメント11_1が図８に示すメインプログラムＰｒｇ＿Ａを実行中に、メインプログラムＰｒｇ＿Ａによって将来的にどのサブプログラムの呼び出しが行われるかを知ることができない。

そのため、図８に示すように、プロセッサエレメント11_1における命令コード「ｇｅｎ（Ｐｒｇ＿Ｄ）」の最初の実行時と２回目の実行時とで、アービタ１６によって、サブプログラムＰｒｇ＿Ｄが異なるプロセッサエレメント11_3，11_4に割り当てられてしまう可能性がある。この場合には、サブプログラムＰ

ｒｇ＿Ｄが比較的短い時間間隔で実行されるにも係わらず、２回目の実行時において、共有メモリ１５からプロセッサエレメント11_3にサブプログラムＰｒｇ＿Ｄを読み込む必要があり、プロセッサエレメント11_3の待ち時間を長引かせている。

このような事態は、図６に示すローカルメモリ３２のメモリ容量と読み込みを行うプログラムの容量とが同じオーダーである場合に特に頻繁に発生し、マルチプロセッサ１の性能を大幅に低下させる要因となっている。

【００２０】

本発明は上述した従来技術の問題点に鑑みてなされ、並列に実行されるプログラム相互間で、多様な形態の同期をとることができる並列処理装置、並列処理方法および当該方法の手順をコンピュータで読み取り可能に記録した記録媒体を提供することを目的とする。

また、本発明は、プロセッサエレメントのローカルメモリと共有メモリとの間でのユーザプログラムの入れ替えに伴うプロセッサエレメントの待ち時間を短縮できる並列処理装置を提供することを目的とする。

【００２１】

【課題を解決するための手段】

上述した従来技術の問題点を解決し、上述した目的を達成するために、本発明の第１の観点の並列処理装置は、プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置であって、一の前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記他の処理手段は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する。

【００２２】

本発明の第１の観点の並列処理装置では、一の処理手段と他の処理手段との間で、双方ともプログラムの実行途中で、プログラム内の待機命令および待機解除命令を用いて、命令レベルで同期がとられる。すなわち、一方のプログラムの実

行終了を伴うことなく、プログラム相互間で同期をとることができる。

【００２３】

また、本発明の第１の観点の並列処理装置は、好ましくは、前記他の処理手段は、同期待ち命令を実行して同期待ち状態になり、前記一の処理手段による前記同期待ち命令に対応する前記待機命令の実行に基づいて前記同期待ち状態を解除する。

これにより、前記他の処理手段における前記待機解除命令の実行より前に前記一の処理手段において待機命令が実行されることが回避される。

【００２４】

また、本発明の第２の観点の並列処理装置は、プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置であって、一の前記処理手段は、待機命令を実行すると前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記他の処理手段は、前記待機解除命令を実行したときに、前記一の処理手段が前記待機状態になっていない場合に、前記一の処理手段が前記待機状態になるまで同期待ち状態になる。

【００２５】

すなわち、本発明の第２の観点の並列処理装置では、前記一の処理手段と前記他の処理手段との間で、プログラム内の待機命令および待機解除命令を用いて、命令レベルで同期がとられる。すなわち、一方のプログラムの実行終了を伴うことなく、プログラム相互間で同期をとることができる。また、前記他の処理手段で処理されるプログラム内に、待機解除命令に対応する同期待ち命令が記述されていなくても、前記一の処理手段が前記待機状態になっていない場合には、前記他の処理手段が前記待機状態になるまで同期待ち状態になる。

【００２６】

また、本発明の第３の並列処理装置は、プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段と、前記プログラムを記憶する第１の記憶手段と、前記複数の処理手段にそれぞれ対応して設け

られ、対応する処理手段で実行されるプログラムを前記第１の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第１の記憶手段に比べてアクセス速度が速い第２の記憶手段とを有する並列処理装置であって、前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記第２の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムを継続して記憶する。

【００２７】

本発明の第３の観点の並列処理装置では、一の前記処理手段が、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する場合に、当該一の処理手段に対応する第２の記憶手段には、前記一の処理手段に供給していたプログラムが継続して記憶される。すなわち、当該プログラムの実行を再開する際に、当該プログラムを再び前記第１の記憶手段から前記第２の記憶手段に読み込む必要がない。

【００２８】

また、本発明の第１の観点の並列処理方法は、プログラムに記述された命令に基づいて少なくとも第１の処理および第２の処理を相互に並列に行う並列処理方法であって、前記第１の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記第２の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する。

【００２９】

また、本発明の第２の観点の並列処理方法は、プログラムに記述された命令に基づいて少なくとも第１の処理および第２の処理を相互に並列に行う並列処理方法であって、前記第１の処理は、待機命令の実行によって前記プログラムに基づ

いた処理を中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記第２の処理は、前記待機解除命令の実行時に、前記第１の処理が前記待機状態になっていない場合に、前記第１の処理が前記待機状態になるまで同期待ち状態になる。

【００３０】

また、本発明の第１の観点の記録媒体は、プログラムに記述された命令に基づいて相互に並列に行われる第１の処理および第２の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって、前記第１の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、前記第２の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する処理である。

【００３１】

さらに、本発明の第２の観点の記録媒体は、プログラムに記述された命令に基づいて相互に並列に行われる第１の処理および第２の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって、前記第１の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第２の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、前記第２の処理は、前記待機解除命令の実行時に、前記第１の処理が前記待機状態になっていない場合に、前記第１の処理が前記待機状態になるまで同期待ち状態になる処理である。

【００３２】

【発明の実施の形態】

以下、本発明の実施形態に係わるマルチプロセッサについて説明する。

第１実施形態

図１は、本実施形態のマルチプロセッサ５１の構成図である。

図１に示すように、マルチプロセッサ５１は、例えば、共有バス１７、処理手段および第２の記憶手段としてのプロセッサエレメント61_1～61_4、第１の

記憶手段としての共有メモリ６５およびプログラム割り付け手段としてのアービタ６６を有する。

マルチプロセッサ５１は、例えば、共有バス１７を介して、プロセッサエレメント61_1～61_4、共有メモリ６５およびアービタ６６を相互に接続したバス結合のアーキテクチャを採用しており、これらの構成要素が単体の半導体チップ内に組み込まれている。

【００３３】

〔構成要素の概要〕

先ず、マルチプロセッサ５１と前述した図５に示すマルチプロセッサ１との共通点および相違点について説明する。

共有バス１７は、前述した図５に示す従来のマルチプロセッサ１の共有バス１７と同じである。

また、プロセッサエレメント61_1～61_4は、前述した図５に示すプロセッサエレメント11_1～11_4と比べて、図６に示すようにプロセッサコア３１およびローカルメモリ３２を有するというハードウェア構成は同じであるが、前述した従来のマルチプロセッサ１では用意されていないプログラム（プロセス）相互間の同期確立のための新たな命令コードを実行するため、当該命令コードの実行に伴う動作が異なる。

また、共有メモリ６５は、図５に示す共有メモリ１５と比べて、ハードウェア構成は同じであるが、記憶しているユーザプログラムの内容が異なる。

さらに、アービタ５６は、プロセッサエレメント51_1～51_4の実行状態（処理の負荷など）を監視し、当該実行状態に基づいて、共有メモリ５５に記憶されているソフトウェアリソースをハードウェアリソースであるプロセッサエレメント51_1～51_4に割り当てる点は、図５に示すアービタ１６と同様である。但し、アービタ５６は、プロセッサエレメント61_1～61_4において前述した新たな命令コードが実行されたときに、図５に示すアービタ１６とは異なる種々の同期確立のための処理およびプロセッサエレメント61_1～61_4へのユーザプログラムの割り当て処理を行う。

【００３４】

〔命令コード〕

マルチプロセッサ５１は、以下に示す命令コードを採用している。

具体的には、マルチプロセッサ５１は、命令コードとして、プログラム実行指示命令としての「ｇｅｎ（ユーザプログラム名）」、同期待ち命令としての「ｗａｉｔ（ユーザプログラム名）」、待機解除命令としての「ｃｏｎｔ（ユーザプログラム名）」、待機命令としての「ｓｌｅｅｐ」およびプログラム終了命令としての「ｅｎｄ」を有する。なお、マルチプロセッサ５１は、その他にも、例えば、汎用のマルチプロセッサにおいて用意さている種々の命令コードを有する。

ここで、命令コード「ｇｅｎ（ユーザプログラム名）」、「ｗａｉｔ（ユーザプログラム名）」および「ｅｎｄ」は前述した図５に示すマルチプロセッサ１が採用している同一名の命令コードと同じであり、命令コード「ｃｏｎｔ（ユーザプログラム名）」および「ｓｌｅｅｐ」はマルチプロセッサ５１が独自に採用している命令コードである。

なお、命令コード「ｇｅｎ（ユーザプログラム名）」、「ｗａｉｔ（ユーザプログラム名）」および「ｃｏｎｔ（ユーザプログラム名）」は、ユーザプログラム名を引数としている。なお、引数の数は任意である。

【００３５】

命令コード「ｇｅｎ（ユーザプログラム名）」は、他のプロセッサエレメント51_1～51_4に、引数「ユーザプログラム名」で指定したユーザプログラムの実行を開始することを指示する命令コードである。

命令コード「ｗａｉｔ（ユーザプログラム名）」は、引数「ユーザプログラム名」で指定したユーザプログラムが、命令コード「ｓｌｅｅｐ」あるいは「ｅｎｄ」を実行するまで、同期待ちを行うことを指示する命令コードである。

命令コード「ｃｏｎｔ（ユーザプログラム名）」は、他のプロセッサエレメント51_1～51_4で実行されている引数「ユーザプログラム名」で指定されたユーザプログラムが待機状態になっているときに、当該待機状態を解除することを指示する命令コードである。

命令コード「ｓｌｅｅｐ」は、ユーザプログラムの実行を一時的に停止して待

機状態となる命令コードである。

命令コード「ｅｎｄ」は、ユーザプログラムの実行を終了する命令コードである。

【００３６】

なお、上述した命令コードは、プログラマがユーザプログラム作成時に、ユーザプログラム内に明示して記述してもよいし、コンパイラが必要に応じて自動的に挿入してもよい。

【００３７】

〔各構成要素の詳細〕

プロセッサエレメント61_1は、図５に示すように、処理手段としてのプロセッサコア３１および第２の記憶手段としてのローカルメモリ３２を有する。

なお、本実施形態では、プロセッサエレメント61_1～61_4が相互に同じ構成である場合を例示するが、本発明では、プロセッサエレメント61_1～61_4は同じ構成をしている必要はなく、例えば、プロセッサコア３１の実行速度やローカルメモリ３２の記憶容量などが異なっていてもよい。

【００３８】

ローカルメモリ３２は、共有バス１７を介して共有メモリ１５から読み出されたユーザプログラムを記憶する。

【００３９】

プロセッサコア３１は、ローカルメモリ３２に記憶されているユーザプログラムの命令コードを順次に読み出して実行する。

また、プロセッサコア３１は、命令コード「ｇｅｎ（ユーザプログラム名）」を実行すると、引数「ユーザプログラム名」によって特定されるユーザプログラムの実行指示を、図１に示す共有バス１７を介してアービタ６６に出力する。

また、プロセッサコア３１は、命令コード「ｗａｉｔ（ユーザプログラム名）」を実行すると、同期待ち状態となり、図１に共有バス１７を介してアービタ６６から、引数「ユーザプログラム名」によって特定されたユーザプログラムの命令コード「ｅｎｄ」あるいは「ｓｌｅｅｐ」が実行されたことを示す通知を入力すると、同期待ち状態を解除して次の命令コードを実行する。

【００４０】

また、プロセッサコア３１は、命令コード「ｃｏｎｔ（ユーザプログラム名）」を実行すると、引数「ユーザプログラム名」によって特定されるユーザプログラムの実行の待機状態を解除する指示を共有バス１７を介してアービタ６６に出力し、処理を中断することなく次の命令コードを実行する。

また、プロセッサコア３１は、命令コード「ｓｌｅｅｐ」を実行すると、当該命令コードを実行したことを共有バス１７を介してアービタ６６に通知すると共に、待機状態となり、その後、共有バス１７を介してアービタ６６から待機状態を解除する指示を入力したときに待機状態を解除して次の命令コードを実行する。

また、プロセッサコア３１は、命令コード「ｅｎｄ」を実行すると、当該命令コードを実行したことを共有バス１７を介してアービタ６６に通知すると共に、プログラムの実行を終了する。

【００４１】

共有メモリ６５は、前述したような命令コード「ｇｅｎ（ユーザプログラム名）」、「ｗａｉｔ（ユーザプログラム名）」、「ｃｏｎｔ（ユーザプログラム名）」、「ｓｌｅｅｐ」および「ｅｎｄ」を含む種々の命令コードを記述した例えばユーザプログラムＰｒｇ＿ａ，Ｐｒｇ＿ｂ，Ｐｒｇ＿ｃ，Ｐｒｇ＿ｄ，Ｐｒｇ＿ｅを記憶している。

【００４２】

アービタ６６は、共有バス１７を介してプロセッサエレメント61_1～61_4から、ユーザプログラムの実行指示を入力すると、当該実行指示によって特定されるユーザプログラムを、共有メモリ６５から、プロセッサエレメント61_1～61_4の実行状態に基づいて例えば負荷が最も小さいプロセッサエレメント61_1～61_4のローカルメモリ３２に読み込む。

また、アービタ６６は、ユーザプログラムの実行の待機状態を解除する指示を入力すると、当該指示によって特定されるユーザプログラムを実行しているプロセッサエレメント61_1～61_4に、共有バス１７を介して、待機状態の解除を示す指示を出力する。

【００４３】

また、アービタ６６は、命令コード「ｓｌｅｅｐ」および「ｅｎｄ」を実行したことを示す通知をプロセッサエレメント61_1～61_4から入力すると、当該命令コードを含むプログラムの実行指示を出力したプロセッサエレメント61_1～61_4に、当該通知を入力したことを通知する。

【００４４】

以下、図１に示すマルチプロセッサ５１のプロセッサエレメント61_1～61_4におけるユーザプログラムの実行過程を追いながら、マルチプロセッサ５１の動作を説明する。

ここでは、図２に示すように、プロセッサエレメント61_1，61_2，61_3，61_4において、ユーザプログラムＰｒｇ＿ａ，Ｐｒｇ＿ｂ，Ｐｒｇ＿ｃ，Ｐｒｇ＿ｄ，Ｐｒｇ＿ｅが実行される場合を例示する。

なお、ユーザプログラムＰｒｇ＿ａ，Ｐｒｇ＿ｂ，Ｐｒｇ＿ｃ，Ｐｒｇ＿ｄ，Ｐｒｇ＿ｅは、例えば、磁気ディスク、磁気テープ、光ディスクおよび光磁気ディスクなどのコンピュータによって読み取り可能な記録媒体から図１に示す共有バス１７に読み込まれている。

【００４５】

先ず、アービタ６６によって、共有メモリ６５からプロセッサエレメント61_1のローカルメモリ３２に、図２に示すユーザプログラムＰｒｇ＿ａが読み込まれる。

そして、プロセッサエレメント61_1のプロセッサコア３１において、ユーザプログラムＰｒｇ＿ａに記述された命令コードが順次に実行される。

具体的には、プロセッサエレメント61_1のプロセッサコア３１において、先ず、命令コード「ｇｅｎ（Ｐｒｇ＿ｂ）」が実行され、当該命令コードの引数で特定されるユーザプログラムＰｒｇ＿ｂの実行指示が、図１に示す共有バス１７を介してアービタ６６に出力される。

そして、アービタ６６によって、共有メモリ６５から共有バス１７を介してプロセッサエレメント61_2のローカルメモリ３２に、ユーザプログラムＰｒｇ＿ｂが読み込まれる。その後、プロセッサエレメント61_2において、ローカルメ

モリ３２に記憶されたユーザプログラムＰｒｇ＿ｂに記述された命令コードが順次に読み出されて実行される。

【００４６】

次に、プロセッサエレメント61_1のプロセッサコア３１において、命令コード「ｇｅｎ（Ｐｒｇ＿ｃ）」および「ｇｅｎ（Ｐｒｇ＿ｄ）」が順次に実行され、前述した命令コード「ｇｅｎ（Ｐｒｇ＿ａ）」の場合と同様の処理を経て、プロセッサエレメント61_3および61_4のプロセッサコア３１において、ユーザプログラムＰｒｇ＿ｃおよびＰｒｇ＿ｄの実行がそれぞれ開始する。

【００４７】

次に、プロセッサエレメント61_1のプロセッサコア３１において、命令コード「ｗａｉｔ（Ｐｒｇ＿ｄ）」が実行され、同期待ち状態となる。

【００４８】

その後、プロセッサエレメント61_4において、ユーザプログラムＰｒｇ＿ｄに記述された命令コード「ｓｌｅｅｐ」が実行され、プロセッサエレメント61_4が待機状態になる。

また、プロセッサエレメント61_4において命令コード「ｓｌｅｅｐ」が実行されたことが、共有バス１７を介してアービタ６６に通知される。そして、アービタ６６に当該通知が入力されると、ユーザプログラムＰｒｇ＿ｄの実行指示を出力したプロセッサエレメント61_1に、当該通知を入力したことがアービタ６６から通知される。

そして、プロセッサエレメント61_1がアービタ６６から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント61_1において次の命令コードが実行される。

【００４９】

次に、プロセッサエレメント61_1のプロセッサコア３１において、命令コード「ｗａｉｔ（Ｐｒｇ＿ｃ）」が実行され、同期待ち状態となる。

【００５０】

その後、プロセッサエレメント61_3において、ユーザプログラムＰｒｇ＿ｃの最後に記述された命令コード「ｅｎｄ」が実行され、プロセッサエレメント６

61_3におけるユーザプログラムＰｒｇ＿ｃの実行が終了する。

また、プロセッサエレメント61_3において命令コード「ｅｎｄ」が実行されたことが、共有バス１７を介してアービタ６６に通知される。

そして、アービタ６６に当該通知が入力されると、ユーザプログラムＰｒｇ＿ｃの実行指示を出力したプロセッサエレメント61_1に、当該通知を入力したことがアービタ６６から通知される。

そして、プロセッサエレメント61_1がアービタ６６から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント61_1において次の命令コードが実行される。

なお、アービタ６６は、前述したように、プロセッサエレメント61_3において命令コード「ｅｎｄ」が実行されたことを示す通知を入力すると、その後、プロセッサエレメント61_3の負荷はなくなったと判断し、プロセッサエレメント61_3のローカルメモリ３２を開放する。

【００５１】

次に、プロセッサエレメント61_1のプロセッサコア３１において、命令コード「ｇｅｎ（Ｐｒｇ＿ｅ）」が実行され、前述した命令コード「ｇｅｎ（Ｐｒｇ＿ａ）」の場合と同様の処理を経て、前述したように開放されたプロセッサエレメント61_3のローカルメモリ３２にユーザプログラムＰｒｇ＿ｅが読み込まれ、プロセッサエレメント61_3のプロセッサコア３１においてユーザプログラムＰｒｇ＿ｅが実行される。

【００５２】

次に、プロセッサエレメント61_1において、命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」が実行され、ユーザプログラムＰｒｇ＿ｄの実行の待機状態を解除する指示が、共有バス１７を介してアービタ６６に出力される。そして、アービタ６６によって、ユーザプログラムＰｒｇ＿ｄを実行しているプロセッサエレメント61_4に、共有バス１７を介して、待機状態の解除を示す指示が出力される。

そして、プロセッサエレメント61_4がアービタ６６から当該指示を入力すると、待機状態が解除され、プロセッサエレメント61_4において次の命令コードが実行される。

なお、プロセッサエレメント61_1では、命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」の実行後に、処理を中断することなく、次の命令コードが実行される。

【００５３】

次に、プロセッサエレメント61_1のプロセッサコア３１において、命令コード「ｗａｉｔ（Ｐｒｇ＿ｄ）」が実行され、同期待ち状態となる。

【００５４】

その後、プロセッサエレメント61_4において、ユーザプログラムＰｒｇ＿ｄの最後に記述された命令コード「ｅｎｄ」が実行され、プロセッサエレメント61_4におけるユーザプログラムＰｒｇ＿ｄの実行が終了する。

また、プロセッサエレメント61_4において命令コード「ｅｎｄ」が実行されたことが、共有バス１７を介してアービタ６６に通知される。そして、アービタ６６に当該通知が入力されると、ユーザプログラムＰｒｇ＿ｄの実行指示を出力したプロセッサエレメント61_1に、当該通知を入力したことがアービタ６６から通知される。

そして、プロセッサエレメント61_1がアービタ６６から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント61_1において次の命令コードが実行される。

なお、アービタ６６は、前述したように、プロセッサエレメント61_4において命令コード「ｅｎｄ」が実行されたことを示す通知を入力すると、その後、プロセッサエレメント61_4の負荷はなくなったと判断し、プロセッサエレメント61_4のローカルメモリ３２を開放する。

【００５５】

次に、プロセッサエレメント61_1のプロセッサコア３１において、命令コード「ｗａｉｔ（Ｐｒｇ＿ｅ）」が実行され、同期待ち状態となる。

【００５６】

その後、プロセッサエレメント61_3において、ユーザプログラムＰｒｇ＿ｅの最後に記述された命令コード「ｅｎｄ」が実行され、プロセッサエレメント61_3におけるユーザプログラムＰｒｇ＿ｅの実行が終了する。

また、プロセッサエレメント61_3において命令コード「ｅｎｄ」が実行され

たことが、共有バス１７を介してアービタ６６に通知される。そして、アービタ６６に当該通知が入力されると、ユーザプログラムＰｒｇ＿ｅの実行指示を出力したプロセッサエレメント61_1に、当該通知を入力したことがアービタ６６から通知される。

そして、プロセッサエレメント61_1がアービタ６６から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント61_1において次の命令コードが実行される。

なお、アービタ６６は、前述したように、プロセッサエレメント61_3おいて命令コード「ｅｎｄ」が実行されたことを示す通知を入力すると、その後、プロセッサエレメント61_3の負荷はなくなったと判断し、プロセッサエレメント61_3のローカルメモリ３２を開放する。

【００５７】

以上説明したように、マルチプロセッサ５１によれば、プログラムの終了を示す命令コード「ｅｎｄ」の他に、プログラムの実行の待機状態を指示する命令コード「ｓｌｅｅｐ」と、当該待機状態を解除する命令コード「ｃｏｎｔ」とを用いることで、異なるプロセッサエレメント61_1～61_4上で実行されているプログラムの命令コード相互間で同期をとることができる。そのため、マルチプロセッサ５１によれば、命令コード相互間で同期とる記述を含むプログラムに基づいた多様な処理を行うことが可能になる。

すなわち、従来のマルチプロセッサ１のように、命令コード「ｅｎｄ」によるユーザプログラムの実行終了を伴わずに、ユーザプログラム相互間で同期をとることが可能になる。

【００５８】

また、マルチプロセッサ５１によれば、図２に示すように、ユーザプログラムＰｒｇ＿ａにおいて、命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」の記述の前に、命令コード「ｗａｉｔ（Ｐｒｇ＿ｄ）」が記述されているため、ユーザプログラムＰｒｇ＿ｄの命令コード「ｓｌｅｅｐ」の実行前に、ユーザプログラムＰｒｇ＿ａの命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」が実行されてしまうことを回避できる。これにより、命令コード「ｓｌｅｅｐ」によるプロセッサエレメント61_4

の待機状態が、プロセッサエレメント61_1における命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」の実行によって確実に解除される。

【００５９】

また、マルチプロセッサ５１によれば、命令コード「ｓｌｅｅｐ」が実行された場合に、当該命令コード「ｓｌｅｅｐ」を実行したことを示す通知をアービタ６６が入力することで、当該命令コード「ｓｌｅｅｐ」を実行したプロセッサエレメント61_1～61_4において当該命令コード「ｓｌｅｅｐ」を含むユーザプログラムの実行が将来再び再開されることをアービタ６６が知ることができる。そのため、アービタ６６によって、当該ユーザプログラムが他のユーザプログラムと入れ替えられてしまうことを回避することが可能になり、ユーザプログラムの読み込み動作の回数を削減して処理時間の短縮が図れる。

具体的には、図２に示す例において、共有メモリ６５からプロセッサエレメント61_4のローカルメモリ３２に、ユーザプログラムＰｒｇ＿ｄを１回のみ読み込めばよく、ユーザプログラムＰｒｇ＿ｄの読み込みによるプロセッサエレメント61_4の待ち時間を短くできる。また、ユーザプログラムＰｒｇ＿ｄの実行を再開するときには、プロセッサエレメント61_1における命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」の実行によって、その旨がプロセッサエレメント61_4に瞬時に通知されることから、高速な応答が要求されるリアルタイム性が問われるユーザプログラムを実行する場合には特に有効である。

すなわち、図８を用いて前述したような、プロセッサエレメント11_1における命令コード「ｇｅｎ（Ｐｒｇ＿Ｄ）」の２回目の実行時に、ユーザプログラムＰｒｇ＿Ｄを再びローカルメモリ３２に読み込むという無駄な動作の発生を回避できる。

【００６０】

上述したような共有メモリ６５からローカルメモリ３２へのユーザプログラムの無駄な読み込みを回避することによる効果は、ローカルメモリ３２のメモリ容量と読み込みを行うユーザプログラムの容量とが同じオーダーである場合に特に顕著である。

【００６１】

第２実施形態

本実施形態のマルチプロセッサは、基本的には、前述した第１実施形態のマルチプロセッサ５１と同じであるが、上述した命令コード「ｃｏｎｔ」の代わりに、待機解除命令として以下に示す命令コード「ｃｏｎｔ＿ａ」を用いる点がマルチプロセッサ５１とは異なる。

すなわち、前述した第１実施形態の命令コード「ｃｏｎｔ」は、一のプロセッサエレメントにおける命令コード「ｃｏｎｔ」の実行が、他のプロセッサエレメントにおける当該命令コード「ｃｏｎｔ」に対応する命令コード「ｓｌｅｅｐ」の実行よりも遅いと当該命令コード「ｓｌｅｅｐ」による当該他のプロセッサエレメントの待機状態を解除できない。このような事態を回避するために、第１実施形態では、例えば、図２に示すように、ユーザプログラムＰｒｇ＿ａには、命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」の記述より前に、ユーザプログラムＰｒｇ＿ｄの命令コード「ｓｌｅｅｐ」の実行によって同期待ちが解除される命令コード「ｗａｉｔ（Ｐｒｇ＿ｄ）」を記述する必要があった。

【００６２】

本実施形態では、第１実施形態の命令コード「ｃｏｎｔ」の代わりに、新たな命令コード「ｃｏｎｔ＿ａ」を用いて、それ以前に命令コード「ｗａｉｔ」を記述することを不要にしている。

命令コード「ｃｏｎｔ＿ａ」は、命令コード「ｃｏｎｔ」と同様に、引数としてユーザプログラム名を指定する。すなわち、命令フォーマットは「ｃｏｎｔ＿ａ（ユーザプログラム名）」となる。

また、図１に示すプロセッサエレメント61_1～61_4のプロセッサコア３１は、命令コード「ｃｏｎｔ＿ａ（ユーザプログラム名）」を実行すると、引数「ユーザプログラム名」によって特定されるユーザプログラムの実行の待機状態を解除する指示を図１に示す共有バス１７を介してアービタ６６に出力する。

プロセッサコア３１は、命令コード「ｃｏｎｔ＿ａ（ユーザプログラム名）」を実行すると、その後、処理を中断することなく、次の命令コードを実行する。

また、プロセッサエレメント61_1～61_4は、アービタ６６から逆同期待ち

指示を入力すると、その後、アービタ６６から逆同期待ち状態を解除する指示を入力するまで逆同期待ち状態となる。

【００６３】

アービタ６６は、ユーザプログラムの実行の待機状態を解除する指示を入力すると、当該ユーザプログラムが待機状態である否かを判断し、待機状態であると判断した場合に、当該待機状態のユーザプログラムが割り当てられたプロセッサエレメント61_1～61_4に、共有バス１７を介して、待機状態の解除を示す指示を出力する。

一方、アービタ６６は、上記判断において、当該ユーザプログラムが待機状態ではないと判断した場合には、ユーザプログラムの実行の待機状態を解除する指示を出力したプロセッサエレメント61_1～61_4に逆同期待ち指示を出力する。その後、アービタ６６は、当該ユーザプログラムから、命令コード「ｓｌｅｅｐ」を実行したことの通知を受けると、上記逆同期待ち状態となっているプロセッサエレメント61_1～61_4に、逆同期待ち解除を示す指示を出力すると共に、当該命令コード「ｓｌｅｅｐ」を実行したプロセッサエレメント61_1～61_4に、共有バス１７を介して待機状態を解除する指示を出力する。

【００６４】

以下、図１に示すプロセッサエレメント61_1～61_4におけるユーザプログラムの実行過程を追いながら、本実施形態のマルチプロセッサの動作を説明する。

ここでは、図３に示すように、図１に示すプロセッサエレメント61_1，61_2，61_3，61_4において、ユーザプログラムＰｒｇ＿ａａ，Ｐｒｇ＿ｂ，Ｐｒｇ＿ｃ，Ｐｒｇ＿ｄ，Ｐｒｇ＿ｅが実行される場合を例示する。

図３に示すユーザプログラムＰｒｇ＿ａａは、図２に示す命令コード「ｃｏｎｔ（Ｐｒｇ＿ｄ）」およびその前の命令コード「ｗａｉｔ（Ｐｒｇ＿ｄ）」の代わりに命令コード「ｃｏｎｔ＿ａ（Ｐｒｇ＿ｄ）」を記述している点が前述したユーザプログラムＰｒｇ＿ａとは異なる。また、図３に示すユーザプログラムＰｒｇ＿ｂ，Ｐｒｇ＿ｃ，Ｐｒｇ＿ｄ，Ｐｒｇ＿ｅは、それぞれ図２に示すユーザプログラムＰｒｇ＿ｂ，Ｐｒｇ＿ｃ，Ｐｒｇ＿ｄ，Ｐｒｇ＿ｅと同じである。

【００６５】

先ず、図３に示すように、プロセッサエレメント61_1において、ユーザプログラムＰｒｇ＿ａａに記述された命令コード「ｇｅｎ（Ｐｒｇ＿ｂ）」，「ｇｅｎ（Ｐｒｇ＿ｃ）」，「ｇｅｎ（Ｐｒｇ＿ｄ）」，「ｗａｉｔ（Ｐｒｇ＿ｃ）」が順に実行される。

これらの処理は、図２を用いて前述した同一名の命令コードの処理と同じである。

【００６６】

そして、命令コード「ｗａｉｔ（Ｐｒｇ＿ｃ）」による同期待ち状態が解除されると、プロセッサエレメント61_1のプロセッサコア３１において、命令コード「ｃｏｎｔ＿ａ（Ｐｒｇ＿ｄ）」が実行される。

これにより、ユーザプログラムＰｒｇ＿ｄの実行の待機状態を解除する指示が、図１に示す共有バス１７を介してプロセッサエレメント61_1からアービタ６６に出力される。

そして、アービタ６６において、プロセッサエレメント61_4で実行されているユーザプログラムＰｒｇ＿ｄが待機状態である否かが判断され、この場合には、待機状態となっていないので、プロセッサエレメント61_1に逆同期待ち指示が出力される。

そして、当該逆同期待ち指示を入力したプロセッサエレメント61_1は、逆同期待ち状態となる。

その後、プロセッサエレメント61_4において、ユーザプログラムＰｒｇ＿ｄの命令コード「ｓｌｅｅｐ」が実行され、当該命令コード「ｓｌｅｅｐ」を実行したことを示す通知が、共有バス１７を介してプロセッサエレメント61_4からアービタ６６に出力される。そして、当該通知に基づいて、共有バス１７を介してアービタ６６からプロセッサエレメント61_1に、逆同期待ち状態を解除する指示が出力され、プロセッサエレメント61_1の逆同期待ちが解除される。

次に、プロセッサエレメント61_1において、ユーザプログラムＰｒｇ＿ａａの命令コード「ｇｅｎ（Ｐｒｇ＿ｅ）」，「ｗａｉｔ（Ｐｒｇ＿ｄ）」，「ｗａｉｔ（Ｐｒｇ＿ｅ）」が実行される。当該実行による処理は、図２を用いて前述

した場合と同じである。

【００６７】

なお、例えば、図４に示すように、プロセッサエレメント61_4における命令コード「ｓｌｅｅｐ」が、プロセッサエレメント61_1における「ｃｏｎｔ＿ａ（Ｐｒｇ＿ｄ）」の実行より早いタイミングで行われた場合には、プロセッサエレメント61_4は、命令コード「ｓｌｅｅｐ」を実行してから待機状態となり、プロセッサエレメント61_1における命令コード「ｃｏｎｔ＿ａ（Ｐｒｇ＿ｄ）」の実行に基づいた待機解除指示をアービタ６６から入力したときに、当該待機状態を解除する。

【００６８】

以上説明したように、本実施形態のマルチプロセッサによれば、上述したような命令コード「ｃｏｎｔ＿ａ」を用いることで、例えば、図２に示す第１実施形態のように、命令コード「ｃｏｎｔ」の記述より前に命令コード「ｓｌｅｅｐ」の実行によって同期待ちが解除される命令コード「ｗａｉｔ」を記述する必要がない。すなわち、例えば、図２に示す場合に、ユーザプログラムＰｒｇ＿ｄの命令コード「ｓｌｅｅｐ」の実行が、命令コード「ｃｏｎｔ＿ａ（Ｐｒｇ＿ｄ）」より前のタイミングで実行された場合でも、プロセッサエレメント61_1が逆同期待ちになり、ユーザプログラムＰｒｇ＿ａａとＰｒｇ＿ｄとの間の同期が保証される。

そのため、プログラマはユーザプログラムＰｒｇ＿ｄの命令コード「ｓｌｅｅｐ」の実行タイミングを意識せずにユーザプログラムＰｒｇ＿ａａを記述でき、プログラマの負担を軽減できる。

【００６９】

本発明は上述した実施形態には限定されない。

例えば、上述した実施形態では、プロセッサエレメント61_1～61_2が、命令コード「ｃｏｎｔ」および「ｃｏｎｔ＿ａ」の実行に応じて出力した待機状態を解除する指示を、アービタ６６を介して、待機状態のユーザプログラムを実行しているプロセッサエレメント61_1～61_4に出力する場合を例示したが、待機状態のユーザプログラムを実行しているプロセッサエレメント61_1～61_4

が共有バス１７を監視し、待機状態を解除する指示をアービタ６６を介さずに入力するようにしてもよい。

また、同様に、プロセッサエレメント61_1～61_4から出力された命令コード「ｓｌｅｅｐ」および「ｅｎｄ」を実行したことを示す通知を、アービタ６６を介さずに、共有バス１７を監視している対応するプロセッサエレメント61_1～61_4が直接入力するようにしてもよい。

【００７０】

また、上述した実施形態では、ユーザプログラムＰｒｇ＿ａ，Ｐｒｇ＿ａａのみに、命令コード「ｇｅｎ」，「ｃｏｎｔ」および「ｃｏｎｔ＿ａ」が記述されている場合を例示したが、これらの命令コードはユーザプログラムＰｒｇ＿ｂ，Ｐｒｇ＿ｃ，Ｐｒｇ＿ｄにも記述されていてもよい。

また、複数のユーザプログラムに、命令コード「ｓｌｅｅｐ」が記述されていてもよい。

【００７１】

また、上述した実施形態では、４個の同一構成のプロセッサエレメントを有するマルチプロセッサを例示したが、プロセッサエレメントの数は２以上であれば任意であり、複数のプロセッサエレメントの構成が異なっていてもよい。

【００７２】

また、上述した実施形態では、図１に示す構成要素が同一半導体チップ内に組み込まれている場合を例示したが、本発明は、例えば、図１に示すプロセッサエレメント61_1～61_4が、ネットワークを介して接続された相互に異なるコンピュータ内にそれぞれ組み込まれた分散処理システムにも適用できる。

【００７３】

【発明の効果】

以上説明したように、本発明の並列処理装置によれば、複数の処理手段で並列に実行されるプログラム相互間で、多様な形態の同期をとることができる。

また、本発明の並列処理方法および記録媒体によれば、並列に行われる複数の処理の相互間で、多様な形態の同期をとることができる。

また、本発明の並列処理装置によれば、処理手段が待機命令を実行することで

、当該処理手段に対応する第２の記憶手段への第１の記憶手段からのプログラムの読み込みを制御でき、第１の記憶手段から第２の記憶手段へのプログラムの読み込み動作の回数を削減し、処理手段の待ち時間を短縮できる。

【図面の簡単な説明】

【図１】

図１は、本発明の第１実施形態のマルチプロセッサの構成図である。

【図２】

図２は、図１に示す本発明の第１実施形態のマルチプロセッサの動作を説明するための図である。

【図３】

図３は、本発明の第２実施形態のマルチプロセッサの動作を説明するための図である。

【図４】

図３は、本発明の第２実施形態のマルチプロセッサのその他の動作を説明するための図である。

【図５】

図５は、一般的なマルチプロセッサの構成図である。

【図６】

図６は、図１および図５に示すプロセッサエレメントの内部構成図である。

【図７】

図７は、図５に示すマルチプロセッサの各プロセッサエレメントへのプログラムの割り付けを説明するための図である。

【図８】

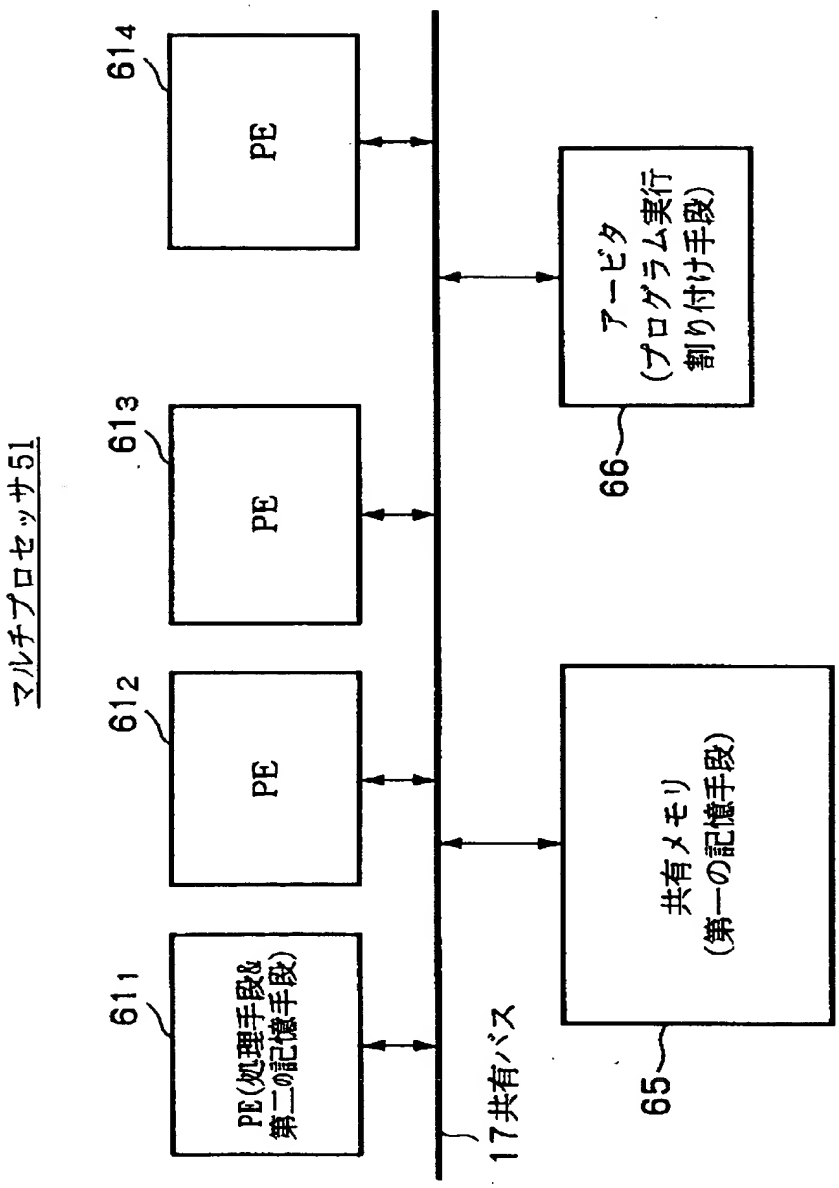
図８は、図５に示す一般的なマルチプロセッサの動作を説明するための図である。

【符号の説明】

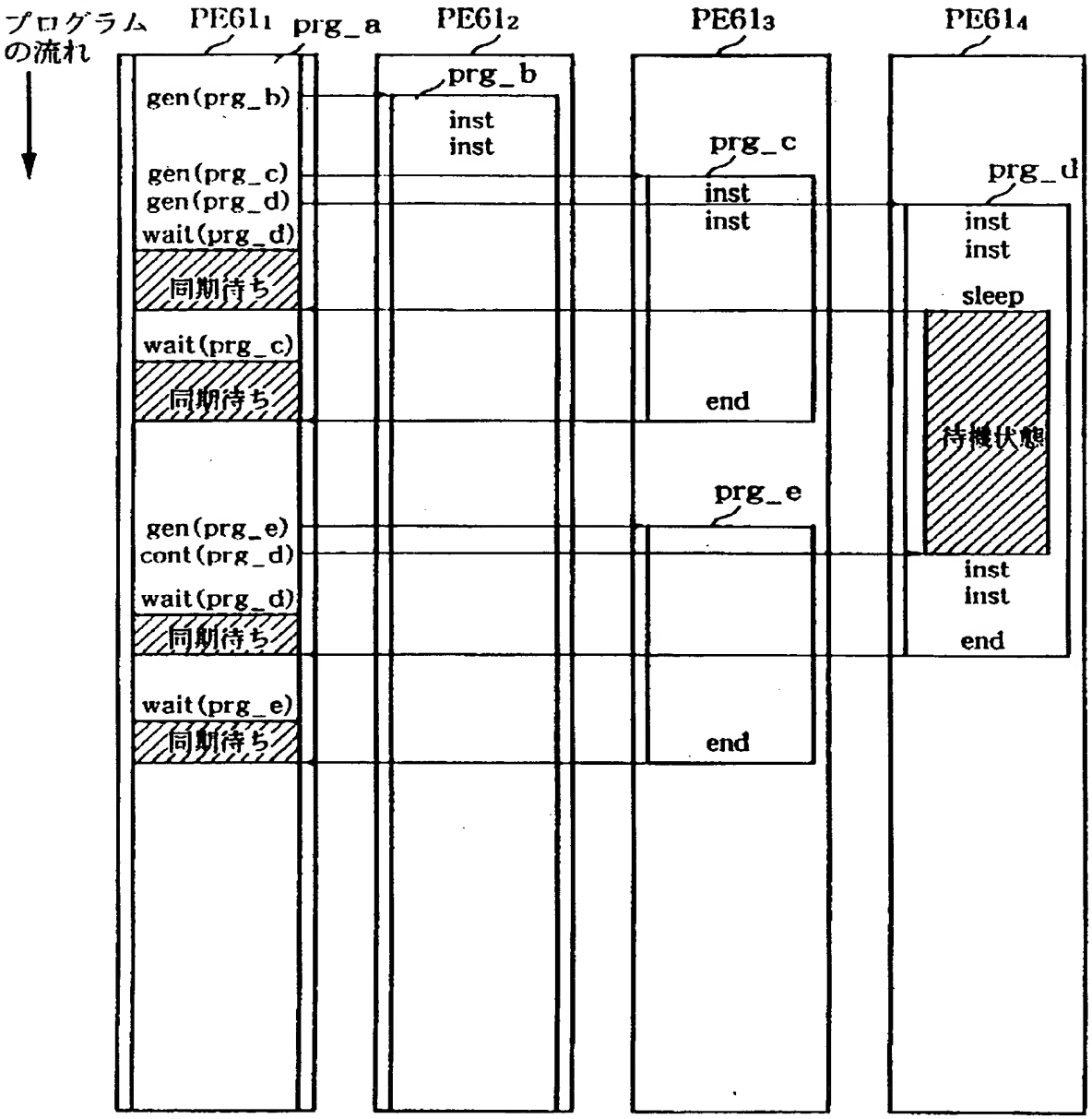
１７…共有バス、３１…プロセッサコア、３２…ローカルメモリ、５１…マルチプロセッサ、61_1～61_4…プロセッサエレメント、６５…共有メモリ、６６…アービタ

【書類名】　図面

【図１】

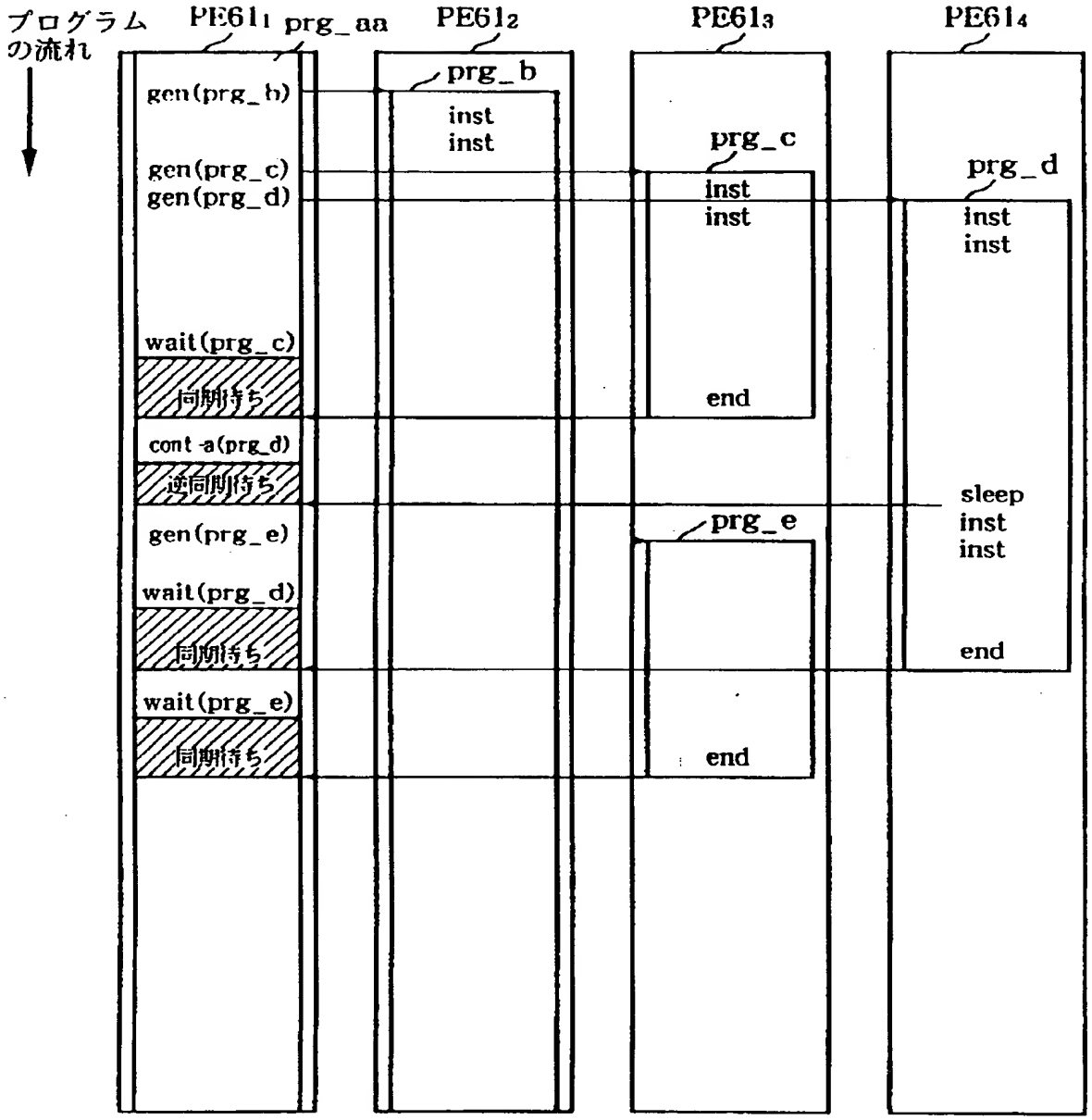

マルチプロセッサ51
614
PE
613
PE
612
PE
611
PE(処理手段&
第二の記憶手段)
17共有バス
アービタ
(プログラム実行
割り付け手段)
66
共有メモリ
(第一の記憶手段)
65

【図２】



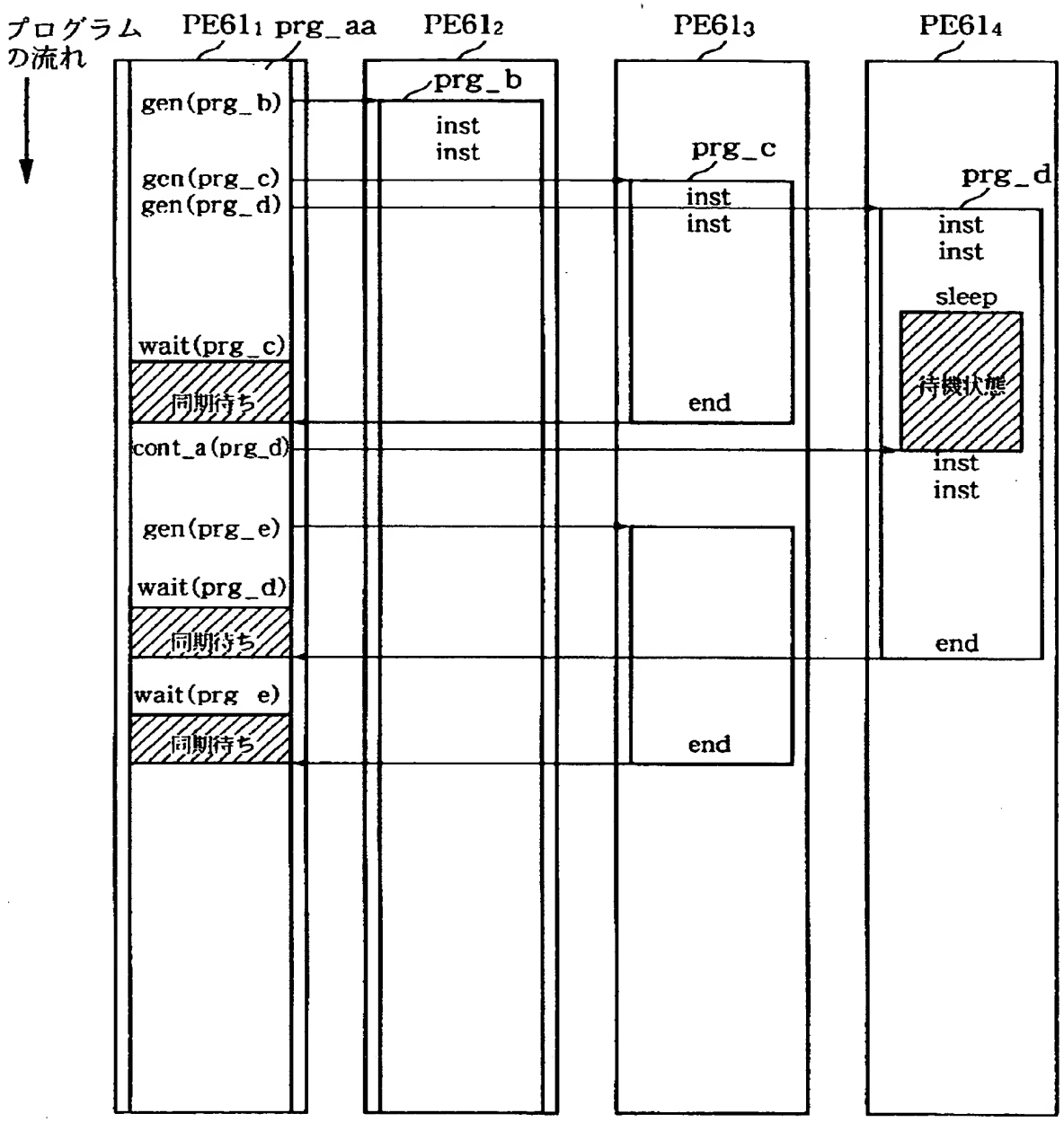
「sleep」：待機命令
「cont」：待機解除命令
「wait」：同期待ち命令
「gen」：プログラム実行指示命令
「end」：プログラム終了命令

【図３】

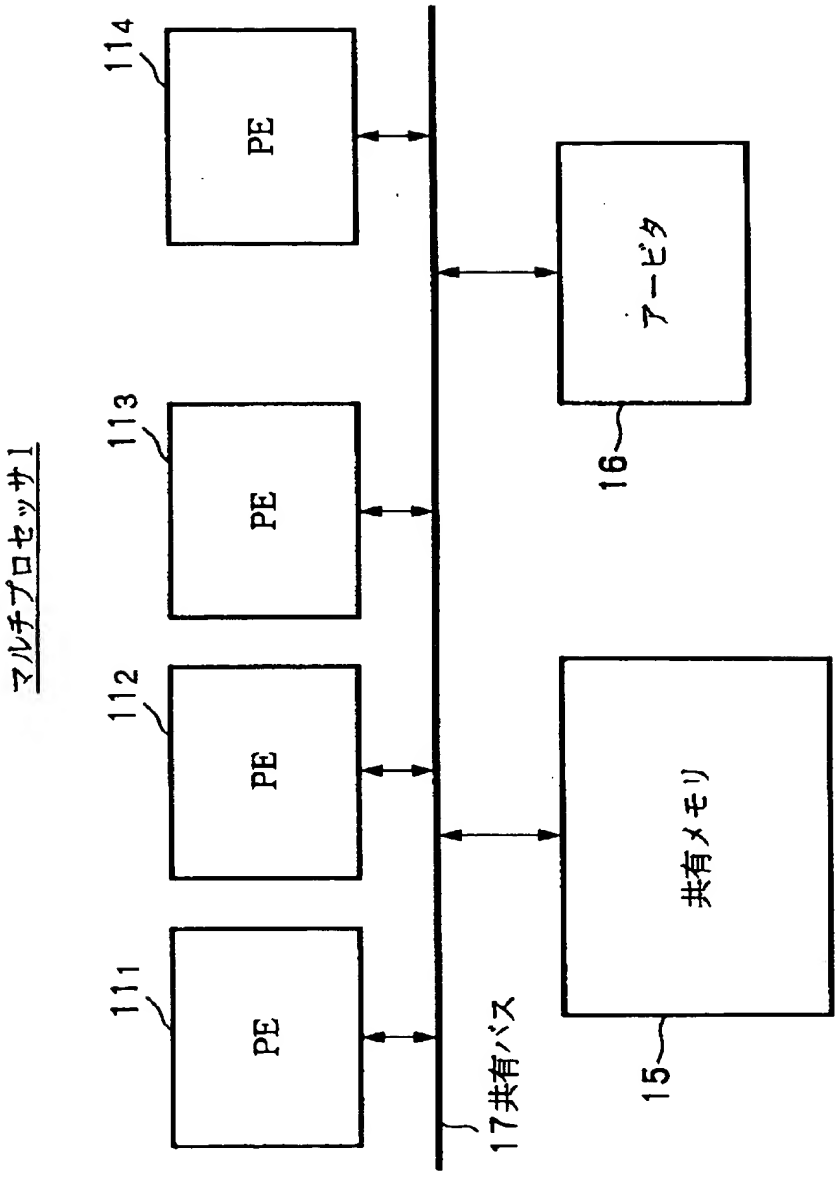


「cont_a」：待機解除命令

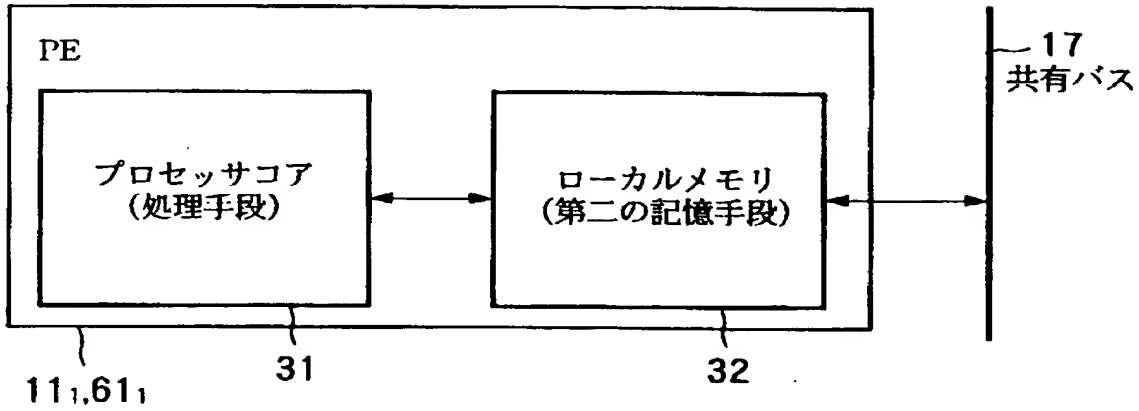
【図４】


プログラムの流れ
PE61₁ prg_aa
PE61₂
PE61₃
PE61₄
prg_b
prg_c
prg_d
gen(prg_b)
inst
inst
gen(prg_c)
gen(prg_d)
inst
inst
inst
inst
sleep
wait(prg_c)
同期待ち
待機状態
end
cont_a(prg_d)
inst
inst
gen(prg_e)
wait(prg_d)
同期待ち
end
wait(prg e)
同期待ち
end

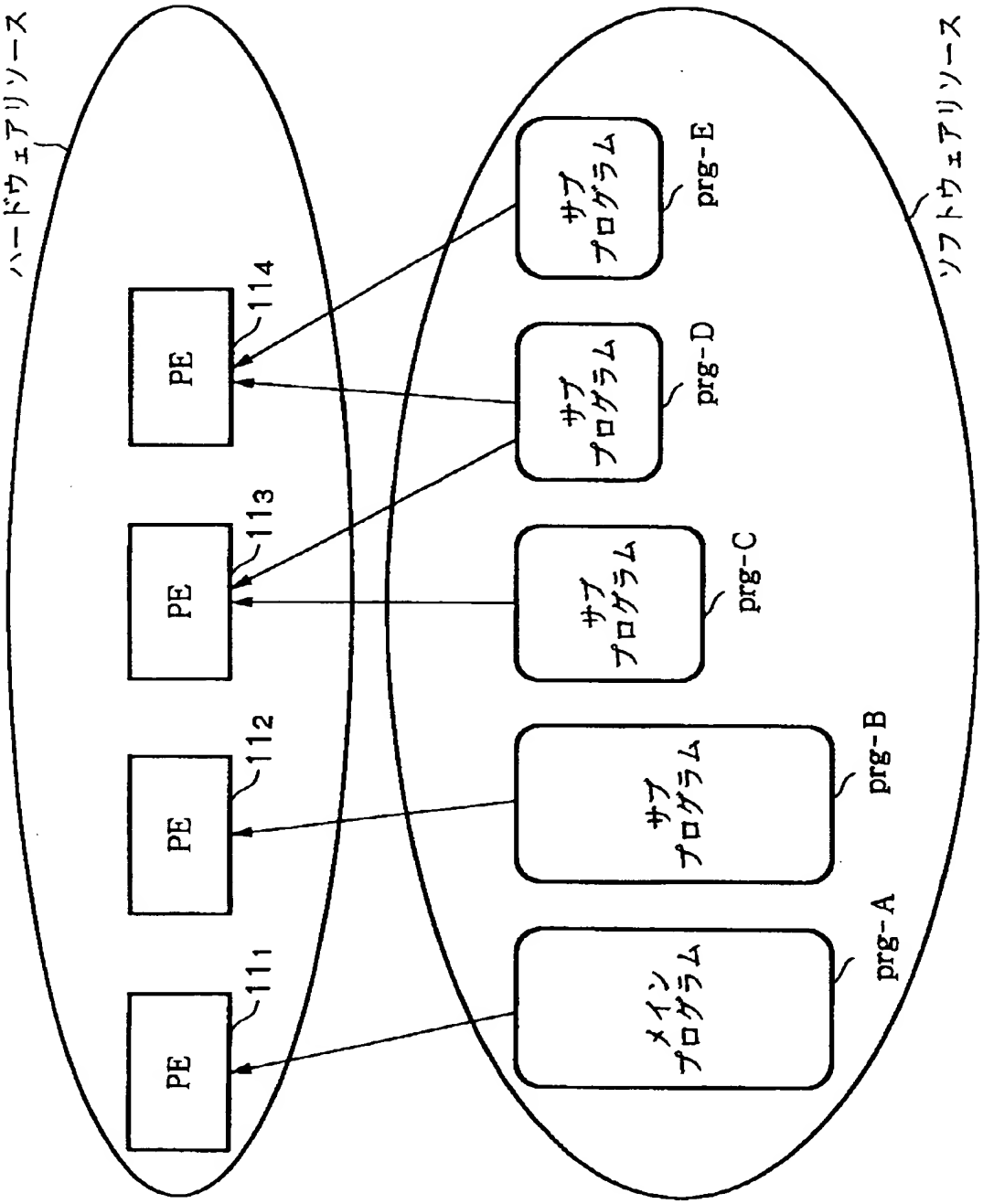
【図５】



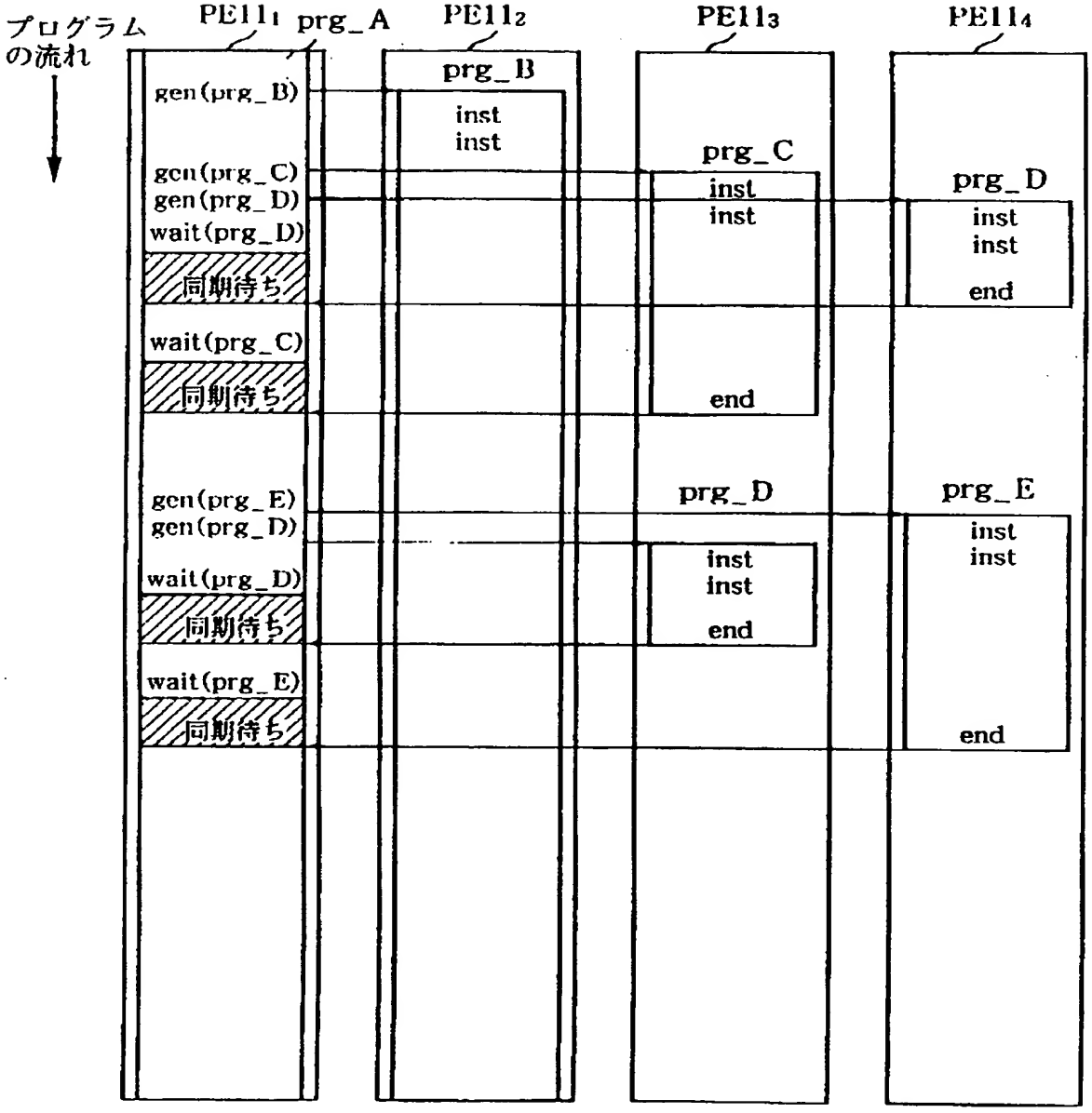
【図６】



【図７】


ハードウェアリソース
PE
111
PE
112
PE
113
PE
114
ソフトウェアリソース
メイン
プログラム
prg-A
サブ
プログラム
prg-B
サブ
プログラム
prg-C
サブ
プログラム
prg-D
サブ
プログラム
prg-E

【図８】



「end」：プログラム終了命令

【書類名】　　要約書

【要約】

【課題】　並列に実行されるプログラム相互間で、多様な形態の同期をとることができる並列処理装置を提供する。

【解決手段】　プロセッサエレメント61_4は、ユーザプログラムＰｒｇ＿ｄにおいて待機命令「ｓｌｅｅｐ」を実行したときに処理を中断して待機状態になり、プロセッサエレメント61_1による待機解除命令「ｃｏｎｔ（Ｐｒｇ＿ｄ）の実行に基づいて前記待機状態を解除して処理を再開し、プロセッサエレメント61_1は、待機解除命令「ｃｏｎｔ（Ｐｒｇ＿ｄ）を実行後、処理を中断することなく次の命令を実行する

【選択図】　図２

【書類名】　職権訂正データ
【訂正書類】　特許願

＜認定情報・付加情報＞
【特許出願人】
【識別番号】　000002185
【住所又は居所】　東京都品川区北品川６丁目７番３５号
【氏名又は名称】　ソニー株式会社
【代理人】　申請人
【識別番号】　100094053
【住所又は居所】　東京都台東区柳橋２丁目４番２号　創進国際特許事務所
【氏名又は名称】　佐藤　隆久

出　願　人　履　歴　情　報

識別番号　　〔０００００２１８５〕

１．変更年月日　　１９９０年　８月３０日
［変更理由］　新規登録
住　所　東京都品川区北品川６丁目７番３５号
氏　名　ソニー株式会社